Wei SHUAI, Xiao-ping CHEN, 2019. KeJia: towards an autonomous service robot with tolerance of unexpected environmental changes. *Frontiers of Information Technology & Electronic Engineering*, 20(3):307-317.
https://doi.org/10.1631/FITEE.1900096

KeJia: towards an autonomous service robot with tolerance of unexpected environmental changes

Key words: Robot; Task planning; Manipulation

Corresponding author: Wei Shuai
E-mail: swwsag@mail.ustc.edu.cn

ORCID: http://orcid.org/0000-0002-8362-4331

Motivation

Compared with traditional industrial robots, domestic service robots in home scenarios have significant differences.

The operating environment for industrial robots is highly structured and customized.

Domestic service robots interact with normal users with touch screens or natural language.

Errors can accumulate during the daily use.

Scenario features

Adaptation to a dynamic environment

While a robot is completing its tasks,
people may disturb or assist the robot, making the environment different from the robot's belief.

Error tolerance

During daily use, there are unexpected errors of slight looseness and misalignment on hardware, especially on the structure of the robot's arm and the pre-calibrated position of sensors.

Main idea

Focusing on the tolerance of unexpected changes:

An answer set programming (ASP) based integrated decision-making strategy can continuously sense the environment, verify the effectiveness of actions, and replan when environmental changes affect the execution of the task.

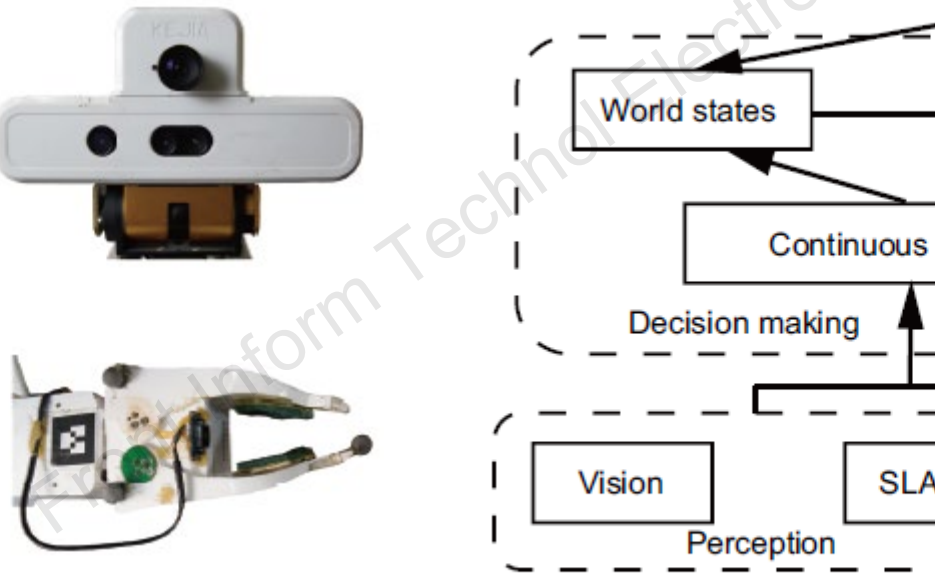
A hierarchical method for KeJia's manipulation is proposed, combining traditional motion planning methods with a learning-based end-to-end approach using a convolution neural network.

A case study is produced when cooking popcorn using an oven.

Method (1/4)

System architecture

(a)



(b)

Fig. 1 Hardware and software architectures of KeJia: (a) hardware structure; (b) software architecture

Method (2/4)

Integrated decision-making

Algorithm 1 Decision-making loop

```
Require: world_states, rules, goals
replan ← true
while replan = true do
    states, actions ← solver(world_states, rules, goals)
    replan ← false
    if states = ∅ and actions = ∅ then
        fail  // no result
    end if
    if states ≠ ∅ and actions = ∅ then
        task is finished
    else
        i ← 0
        while i <len(actions) do
            belief_states ← states[i]
            update percept_states from perception
            if conflict(belief_states, percept_states) the
                world_states ← update(percept_states)
                replan ← true
                break while
            end if
            result ← execute_action(actions[i])
            if result = true then
                world_states ← update(states[i + 1])
                i ← i + 1
            else
                replan ← true
                break while
            end if
        end while
        if replan = false then
            task is finished
        end if
    end if
end while
```

Method (3/4)
Manipulation

Algorithm 2 Manipulation

```
Require: obstacles, robot_model, retry_limit
retry ← 0
while retry < retry_limit do
    trajactories ← planning(obstacles, robot_model)
    if trajectories ≠ ∅ then
        i_model ← inflact(robot_model, error)
        if isValid(obstacles, i_model) then
            execute trajectories
        else
            execute trajectories to blocking location p
            if inRange(p) then
                enter the end-to-end control loop
            else
                percept again and reduce error
                error ← new_error
                retry ← retry + 1
            end if
        end if
    else
        fail
        break while
    end if
end while
```

Motion:
Grasping, placing, and touching

Step 1:
Moving the end effector of the robot's arm to a certain pose

Step 2:
Closing or opening its paw at the previous pose

Task:
Reaching the target within a certain range and not colliding with other obstacles during the movement

Method (4/4)

End-to-end layer

Three important assumptions:

1. During the gripping process, the object being manipulated and the gripper are both in the field of view.

2. After the planning level is executed, a certain error exists between the gripper and the target item.

3. Robot's arm runs with local stability, meaning the joints remain stable in a localized range where there is no applied load or a sudden change in speed.

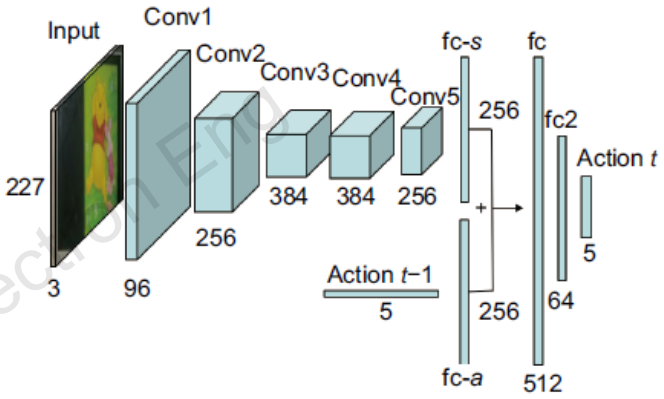


Fig. 3 Network structure

Algorithm 3 End-to-end control loop

```
a_{t-1} ← [1,0,0,0,0]  // action: move forward
loop
    obtain current image I_t
    a_t ← g(I_t, a_{t-1})
    action ← max(a_t)
    execute action
    a_t ← a_{t-1}
    if collision is detected then
        stop
        break loop
    end if
end loop
if the target pose is reached then
    close/open gripper
else
    return fail
end if
```

Major results (1/3)

Simulation of the end-to-end layer

Table 1 Grasp feasibility of different methods with 200 attempts

Method	Success times				Total success times
	$d \in [0,2)$ mm	$d \in [2,3)$ mm	$d \in [3,4)$ mm	$d \in [4,5)$ mm	
Full method	112	3	63	22	200
Without background randomization	101	0	0	92	193
Without gripper in the field of view	0	0	1	41	42
Without camera randomization	0	0	88	86	174

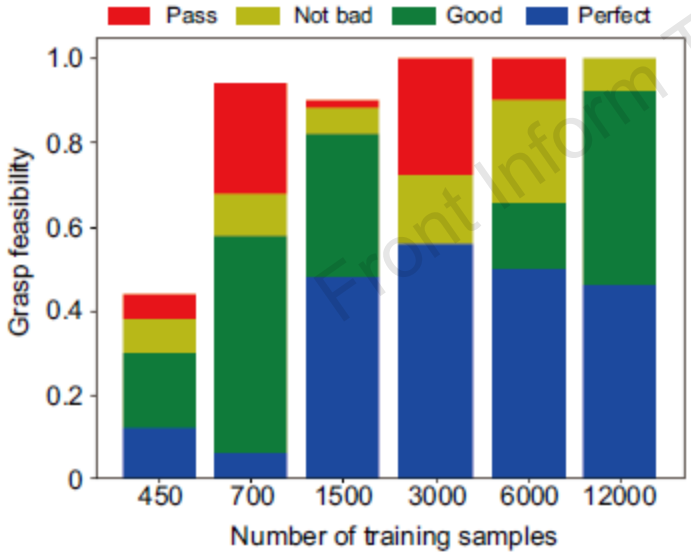


Fig. 4 Relationship between the number of samples and grasp quality

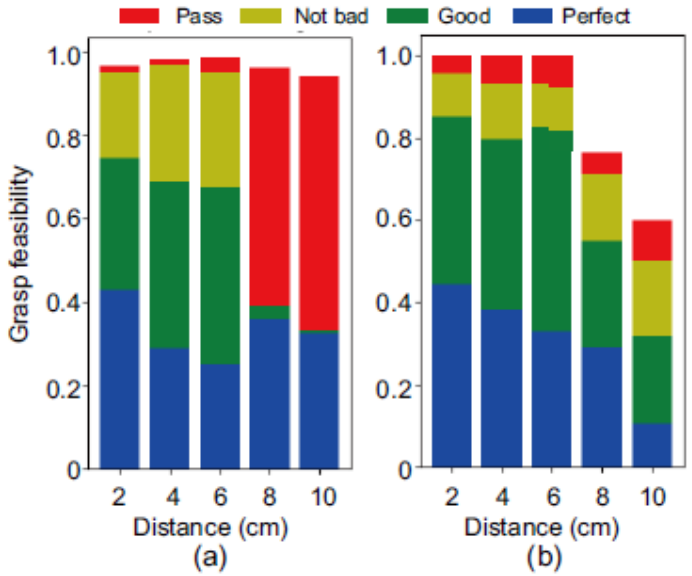


Fig. 5 Relationship between grasp quality and the start position with (a) or without (b) pre-trained weights

Major results (2/3)

Manipulation in the real world

Table 2 Success times in attempts of pressing button

Method	Success times		
	First 15 attempts	Second 15 attempts	Third 15 attempts
Open-loop	15	3	7
Closed-loop	14	15	12
End-to-end	–	–	–
Ours	15	14	15

Major results (3/3)

Case study

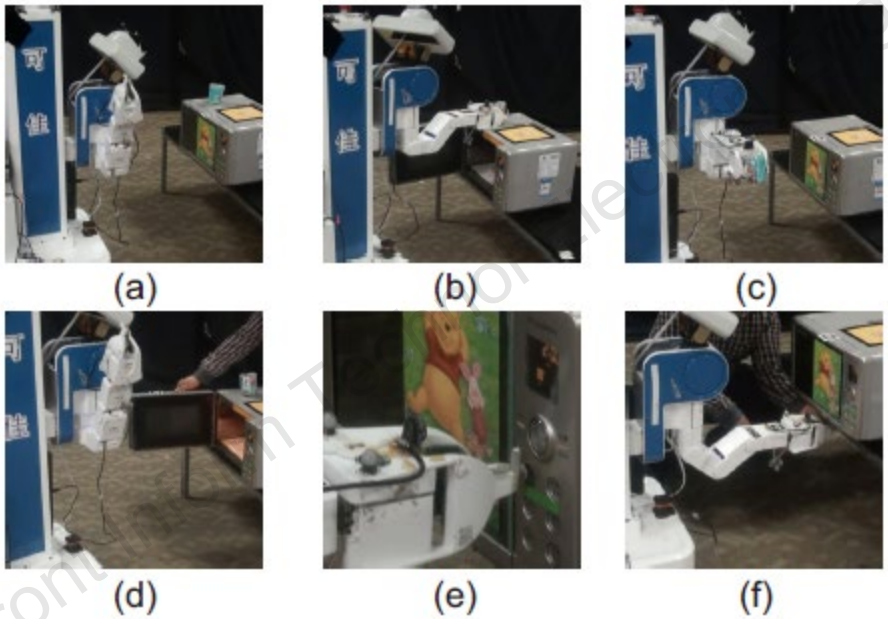


Fig. 6 Scenarios in demonstration: (a) scenario 1; (b) scenario 2; (c) scenario 3; (d) scenario 4; (e) scenario 5; (f) scenario 6

Our demo video can be found at
http://ai.ustc.edu.cn/media/kejia_oven.mp4

Conclusions

Our work has focused on integrated decision making and error-tolerant manipulation. We have developed an intelligent service robot system which can flexibly handle a dynamic environment and tolerate errors during manipulation.

Using the solver based on the ASP and continuous observation, KeJia can adapt itself to some unexpected changes in the environment.

We have proposed a novel hierarchical method which combines motion planning with neural network prediction to enable robots to use low precision equipment to complete high-precision actions. Instead of collecting a large amount of data for a long time, only a simple one-minute data collection is needed for each operating object, which greatly reduces the cost of data collection in robotic applications.